Tutorials Algorithms K-Means clustering in OpenCV

K-Means clustering in OpenCV

By Utkarsh | Published: August 10, 2010

K-Means is an algorithm to detect clusters in a given set of points. It does this without you supervising or correcting the results. It works with any number of dimensions as well (that is, it works on a plane, 3D space, 4D space and any other finite dimensional spaces). And OpenCV comes with this algorithm built right into it!

K-means with OpenCV’s C++ interface

The function you need to call to execute the algorithm is:

```
double kmeans(const Mat& samples,
              int clusterCount,
              Mat& labels,
              TermCriteria termcrit,
              int attempts,
              int flags,
              Mat* centers)
```

This function is in the *cv* namespace. So you can use it by *cv::kmeans* or by simply including the *cv* namespace. If you know how K-means works, the parameters should be self explanatory.

Parameters

- **samples**: *(input)* The actual data points that you need to cluster. It should contain exactly one point per row. That is, if you have 50 points in a 2D plane, then you should have a matrix with 50 rows and 2 columns.

- **clusterCount**: *(input)* The number of clusters in the data points.
- **labels**: *(output)* Returns the cluster each point belongs to. It can also be used to indicate the initial guess for each point.
- **termcrit**: *(input)* This is an iterative algorithm. So you need to specify the termination criteria (number of iterations & desired accuracy)
- **attempts**: *(input)* The number of times the algorithm is run with different center placements
- **flags**: *(input)* Possible values include:
  - **KMEANS_RANDOM_CENTER**: Centers are generated randomly
  - **KMEANS_PP_CENTER**: Uses the kmeans++ center initialization
  - **KMEANS_USE_INITIAL_LABELS**: The first iteration uses the supplied *labels* to calculate centers. Later iterations use random or semi-random centers (use the above two flags for that).
- **centers**: *(output)* This matrix holds the center of each cluster.

Returns

The function returns the compactness of the final clustering. What is compactness? It’s a measure of how good the labeling was done. The smaller the better.

When *attempts* is 1, the value returned is the compactness of the only iteration that happened. If *attempts* is more than 1, the final labeling returned is the one with the least compactness.

K-means with OpenCV’s C interface

The C equivalent of the k-means function is:

```
int cvKMeans2(const CvArr* samples,
              int nclusters,
              CvArr* labels,
              CvTermCriteria termcrit,
              int attempts=1,
              CvRNG* rng=0,
              int flags=0,
              CvArr* centers=0,
              double* compactness=0)
```

The parameters are similar to the C++ interface.

Parameters

- **samples**: *(input)* The actual data points that you need to cluster. It should contain exactly one point per row.

- **nclusters**: *(input)* The number of clusters in the data points.
- **labels**: *(output)* Returns the cluster each point belongs to. It can also be used to indicate the initial guess for each point.
- **termcrit**: *(input)* This is an iterative algorithm. So you need to specify the termination criteria (number of iterations & desired accuracy)
- **attempts**: *(input)* The number of times the algorithm is run with different center placements
- **rng**: (input) A random number generate used to generate the initial guess. Puts you in total control of what's happening.
- **flags**: *(input)* Possible values include:
  - **0**: (the number 0) Centers are generated randomly
  - **KMEANS_USE_INITIAL_LABELS**: The first iteration uses the supplied *labels* to calculate centers. Later iterations use random or semi-random centers (use the above two flags for that).
- **centers**: *(output)* This matrix holds the center of each cluster.
- **compactness**: *(output)* Holds the compactness of the best labeling scheme.

If you're still using the C interface, I highly recommend you shift to the more intuitive and no-more-tears C++ interface!

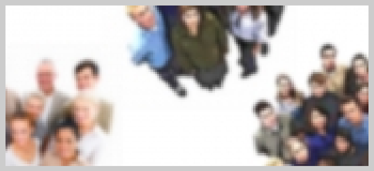
Summary

You got to know how to run K-means without writing any code! You got to know about the C++ and C functions that you can use to execute K-Means on your data sets.

Issues? Suggestions? Visit the Github issue tracker for AI Shack

Back to top

Like 2,492 people like this. Sign Up to see what your friends like.



K-Means clustering

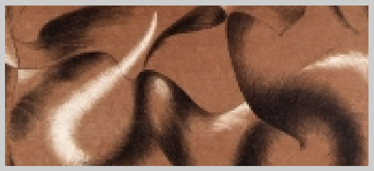


Image Convolutions in OpenCV

20 Comments

Gautam
Posted February 14, 2011 at 12:39 pm | *Permalink*

KMeans doesn’t work with CV_64F Mat?

Reply

Utkarsh
Posted February 14, 2011 at 9:28 pm | *Permalink*

It should work. Haven’t tested it though.

Reply

Amey
Posted March 8, 2011 at 3:00 pm | *Permalink*

Hi Utakarsh

Ur Pages are really awesome..It have helped many peaople a lot for understanding OpenCV. I wanted to know how to apply kmeans2 function in openCV on images.. I am not getting a proper guidance..

Let me know plz

Regards

Reply

Utkarsh
Posted March 9, 2011 at 9:52 am | *Permalink*

What’s the problem? Oh, and, you don’t apply kmeans on an image. You apply it to a dataset – a set of coordinates in some n-dimensional space.

Reply

amey
Posted March 9, 2011 at 7:31 pm | *Permalink*

Thanks for ur reply

Ya..Its applied on a dataset..
But i need to use to on an image for its segmentation…
I want to apply the kmeans on a image..treating it as a dataset…
So any wat to do so

I hope i am clear on my problem..
Regards

Reply

Utkarsh
Posted March 9, 2011 at 7:36 pm | *Permalink*

I doubt if that's possible. You need some way of converting the image into a dataset. You could use each pixel's RGB triplet – and use those to figure out clusters.

Reply

amey
Posted March 9, 2011 at 7:43 pm | *Permalink*

Actually I had read a article which had use k-means for image segmentation..
So I wanted to implement it using OpenCV..
I'll try out something as u suggested…
Will try first on gray-scale image..looks simpler..

Will let u know about it 😀

Thanks a lot for ur assistance

Regards

Reply

Utkarsh
Posted March 9, 2011 at 7:45 pm | *Permalink*

Do you have a link to that article?

Reply

Faiz
Posted April 20, 2011 at 2:27 am | *Permalink*

Do you have an example of coding for the k means c++ interface?

Reply

AruniRC
Posted April 26, 2011 at 10:29 pm | *Permalink*

Hey there!
This article was really helpful but I have some queries. Suppose I am clustering the pixels in a greyscale image. Then for each pixel in CvArr* samples (i convert the 2d image into a 1D array) the corresponding value in CvArr* label will indicate the color it should be set to. Then what is the difference between label and cluster-center?

Reply

Utkarsh
Posted April 29, 2011 at 9:09 pm | *Permalink*

You cannot cluster the pixels in a greyscale image. You can either cluster the pixels' locations (maybe based on their intensity) or cluster intensity in the image.

And I think the label and cluster center refer to the same thing.

Reply

AruniRC
Posted April 30, 2011 at 5:21 am | *Permalink*

Sorry I was trying to indicate clustering the pixel intensity. In short if there are say 0-255 colors present the clustering would result in say 6 greylevels (taking k=6) thus reducing the no. of colors present.

Reply

Utkarsh
Posted April 30, 2011 at 7:15 am | *Permalink*

Oh okay.

Reply

Srikanta

Posted May 20, 2011 at 5:47 pm | *Permalink*

I am working on blob tracking. Now I want to extract the blobs from color image. Is it possible to make cluster the color image by opencv function cvKMeans(). If possible then what sholud be the parameter values. Suppose I have the image IplImage *colorImage and I want 10 cluster. Thanking you.

Reply

Utkarsh

Posted June 17, 2011 at 4:51 pm | *Permalink*

Na – you're looking for blob tracking. Not clustering.

Reply

AruniRC

Posted June 4, 2011 at 10:23 am | *Permalink*

The OpenCV samples show hot to do clustering on a random 2D point set – the code is quite confusing.
I'm trying to cluster a set of data in a floating-point array e.g. – val[] = {12.5, 5.6, 14.2, 3.4, 20.5, 2.9, 3.1};
could you please let me know how to get this data into a CvMat* and then do the clustering with no. of clusters = 2 ? Using the C interface.

Reply

Utkarsh

Posted June 16, 2011 at 7:13 pm | *Permalink*

Figured it out yet? You need to make a CvMat with 1 column.

Reply

sotiraw
Posted June 26, 2011 at 8:23 pm | *Permalink*

can you give an example with sift and k means together? so it really shows the use of it?
find sift keypoints with opecvs sift an them cluster them

Reply

nikita
Posted August 2, 2011 at 2:28 pm | *Permalink*

yes ,what sotiraw says would be a great example

Reply

Utkarsh
Posted August 9, 2011 at 7:22 pm | *Permalink*

Hmm. Sounds interesting. Let’s see if I can make something like that.

Reply

Post a Comment

Your email is *never* published nor shared. Required fields are marked *

Name *

Email *

Website

Comment

You may use these HTML tags and attributes `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>`

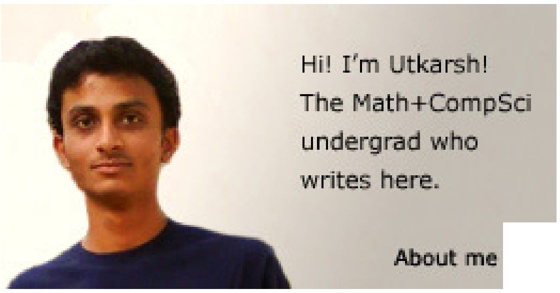
Post Comment

Free updates

Email Address

More info

1000+ subscribers!



Popular Posts

- Scanning QR Codes
- The Canny Edge Detector
- SuDoKu Grabber with OpenCV
- Implementing Canny Edges from scratch
- Image Moments

More on AI SHack

Popular **Beginners** **Utkarsh's favs**

Scanning QR Codes

The Canny Edge Detector

SuDoKu Grabber with OpenCV

Implementing Canny Edges from scratch

Image Moments

About me

My name is Utkarsh Sinha, and I'm an undergraduate student, pursuing B.E. Computer Science + M.Sc. Mathematics. Here, I help you understand ideas in Artificial Intelligence, using a not so techy and mathematical language. And in the process, learn more about Artificial Intelligence myself.

Read more at the about page

Created by Utkarsh Sinha - d312a67 - Linode